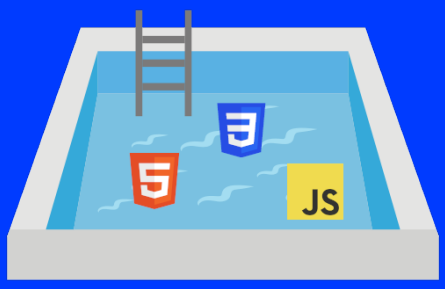
{EPITECH}

WEB DEV POOL_

< DAY 06 - JAVASCRIPT />



WEB DEV POOL

The next few days will be dedicated to the JavaScript language. Let's get comfy with it.

This day is tested by the autograder! We will use `bun` to run your JavaScript files.

If you need a little break in your day or later, we strongly suggest you to:

- ✓ play the cyberpunk text-based incremental RPG bitburner ;
- ✓ collect coins in the platform game JSRobot ;
- ✓ code your way through the dungeons of warriorjs ;
- ✓ program the elevators' movements in Elevator saga ;
- ✓ solve the mazes of Untrusted, the katas of jskatas, the JSDares puzzles.

Task 01

Delivery: `task01.js`
Prototype: `drawTriangle(height)`

Write a function `drawTriangle` that:

- ✓ takes a triangle height as parameter ;
- ✓ draws a triangle on the standard output ;
- ✓ is exported and contained in a `task01.js` file.

Terminal

```
$>  cat task01.js
export function drawTriangle(height) {
//
// your code here
//
}
```

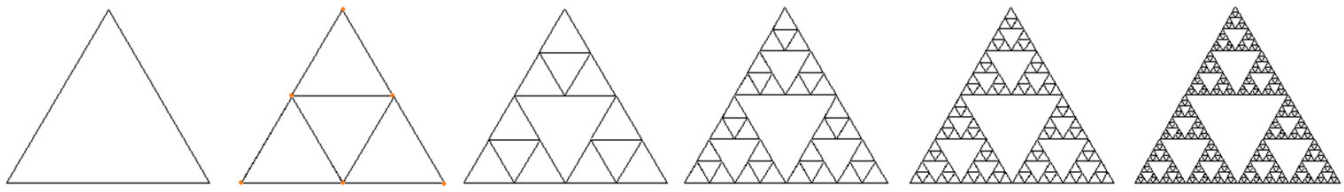
Your function will be tested the following way:

Terminal

```
$>  cat task01_tester.js
import { drawTriangle } from './task01.js';
drawTriangle(5);
```

Terminal

```
$>  bun task01_tester.js
$
$$
$$$
$$$$
$$$$$
```



Task 02

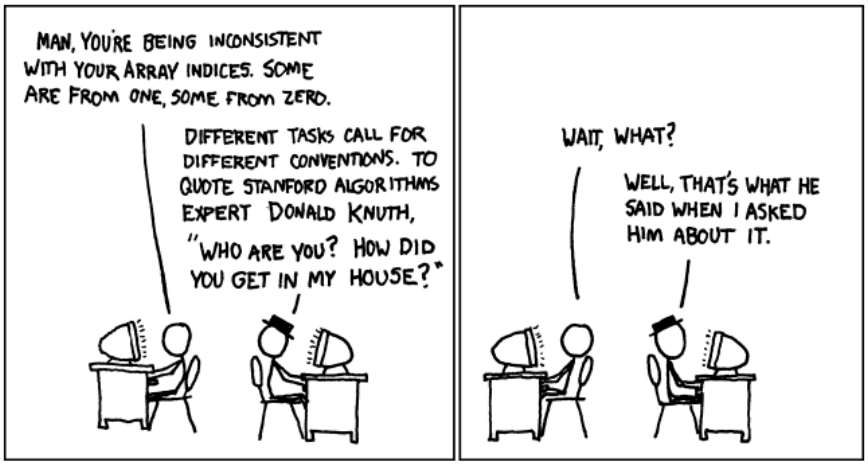
Delivery: `task02.js`
Prototype: `arraysAreEqual(arr1, arr2)`

Write a function `arraysAreEqual` that:

- ✓ takes two arrays as arguments ;
- ✓ returns true if both arrays are equal, false otherwise ;
- ✓ is exported and contained in a `task02.js` file.

Terminal

```
$>  cat task02.js
export function arraysAreEqual(arr1, arr2) {
// your code here
}
```



Your function will be tested the following way:

Terminal

```
$>  cat task02_tester.js
import { arraysAreEqual } from './task02.js';
console.log(arraysAreEqual([1, 2], [1, 4]) ? 'True' : 'False');
```

Terminal

```
$>  bun task02_tester.js
False
```

Task 03

Delivery: `task03.js`
Prototype: `countGs(str)`

Write a `countGs` function that:

- ✓ takes a string as parameter ;
- ✓ returns the number of 'G' it contains ;
- ✓ is exported and contained in a `task03.js` file.

Task 04

Delivery: `task04.js`
Prototype: `fizzBuzz(num)`

Write a `fizzBuzz` function that:

- ✓ takes a number as parameter ;
- ✓ prints all the numbers from 1 to this number ;
- ✓ is exported and contained in a `task04.js` file ;
- ✓ complies with the following requirements:
  1. print ”Fizz” instead of the number if it is divisible by 3 ;
  2. print ”Buzz” instead of the number if it is divisible by 5 ;
  3. print ”FizzBuzz” instead of the number if it divisible by both 5 and 3.

The output terms (be it a number or a string) should be comma separated.

Task 05

Delivery: `task05.js`
Restriction: only ES5 is allowed
Prototype: `range(start, end, step)`

Write a `range` function that:

- ✓ takes 3 integers named `start`, `end` and `step` as arguments ;
- ✓ returns an array containing all the numbers from `start` up to `end` included ;
- ✓ with an increment corresponding to the optional third argument `step` ;
- ✓ is exported and contained in a `task05.js` file.

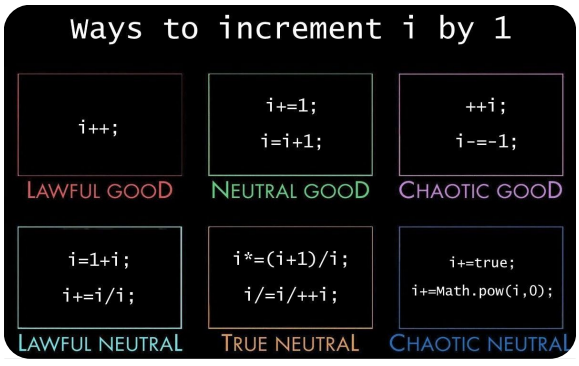
If `step` is not provided, then increment = 1.
You code should cover all types of integers!

Terminal

```
$>  cat example.js
// ...
console.log(range(1, 10, 2));
console.log(range(19, 22));
console.log(range(5, 2, -1));
```

Terminal

```
$>  bun example.js
[ 1, 3, 5, 7, 9 ]
[ 19, 20, 21, 22 ]
[ 5, 4, 3, 2 ]
```



Task 06

Delivery: `task06.js`
Prototype: `objectsDeeplyEqual(cmp1, cmp2)`

In `task06.js`, write the function `objectsDeeplyEqual` that takes two arguments and returns true only if:

- ✓ the arguments have the same value ;
- ✓ or they are objects with the same properties whose values are equal when compared with a **recursive** call to `objectsDeeplyEqual` ;

Terminal

```
$>  cat example.js
// ...
const obj = {here: {is: "an"}, object: 2};
console.log(objectsDeeplyEqual(obj, obj));
console.log(objectsDeeplyEqual(obj, {here: 1, object: 2}));
console.log(objectsDeeplyEqual(obj, {here: {is: "an"}, object: 2}));
```

Terminal

```
$>  bun example.js
true
false
true
```

Your function should figure out whether to compare two things by identity or by looking at their properties, and it's not supposed to be too complex (this is only the first day of JS).

'null' is also an ”object”.

Non-zero value

0

null

undefined

Task 07

Delivery: `task07.js`
Prototype: `arrayFiltering(array, test)`

In `task07.js` , write the `arrayFiltering` function that:

- ✓ takes two arguments: `array` and `test` (where `test` is a function returning a boolean) ;
- ✓ calls the `test` function for each element contained in `array` ;
- ✓ returns a new array, containing the filtered values for which `test` returned True.

Terminal

```
$>  cat example.js
// ...
const toFilter = [1, 2, 3, 4, 5, 6, 7, 8, 9];
const res = arrayFiltering(toFilter, function (value) {
return value % 3 === 0;
});
console.log(res);
```

Terminal

```
$>  bun example
[3, 6, 9]
```

Your function should NOT care about the implementation of the `test` function.

v 4.2